Obliczenia Naukowe

Lista 1

Mateusz Kościelniak
nr indeksu: 244973

Zadanie 1

- **Opis problemu**

  Powtórzenie zadania 5 z listy 1 polegającego na mnożeniu wektorów różnymi algorytmami po modyfikacji wektorów.

- **Rozwiązanie**

  Do rozwiązania zastosowałem algorytmy mnożenia zaimplementowane przeze mnie w 5 zadaniu listy 1. Modyfikacji uległy jedynie wektory w taki oto sposób, że z wektora $x = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$ usunąłem ze składowych czwartej i piątej po jedną ostatnią cyfrę dziesiętną, w ten sposób powstał wektor $x' = [2.718281828, -3.141592654, 1.414213562, 0.577215664, 0.301029995]$.

- **Wyniki**

  Tabela, z wynikami przed zmianą wektorów (x, y) i po zmianie (x', y) .

algorytm	x, y	x', y
Float32		
1	-0.4999443	-0.4999443
2	-0.4543457	-0.4543457
3	-0.5	-0.5
4	-0.5	-0.5
Float64		
1	1.0251881368296672e-10	-0.004296342739891585
2	-1.5643308870494366e-10	-0.004296342998713953
3	0.0	-0.004296342842280865
4	0.0	-0.004296342842280865

- **Wnioski**

  Możemy zauważyć, że dla arytmetyki Float32 błędy obliczeń, są takie same dla wektora oryginalnego i po zmianie, jest to spowodowane tym, że precyzja arytmetyki jest za mała aby uwzględnić wprowadzoną zmianę., tzn. usunięcie najmniej znaczących cyfr składowych wektora x nie zmieniło sposobu zapisu liczby w arytmetyce Float32. W arytmetyce Float64 wyniki zmieniły się, ponieważ ma ona większą precyzję. Dane były dobrane w ten sposób, abyśmy mogli zobaczyć, jak duże znaczenie ma dobór precyzji arytmetyki, w implementacji

algorytmów. Widać tutaj również, że niewielkie zmiany powodują względnie duże odkształcenia wyników, co świadczy o tym, że zadanie jest źle uwarunkowane.

Zadanie 2

- **Opis problemu**

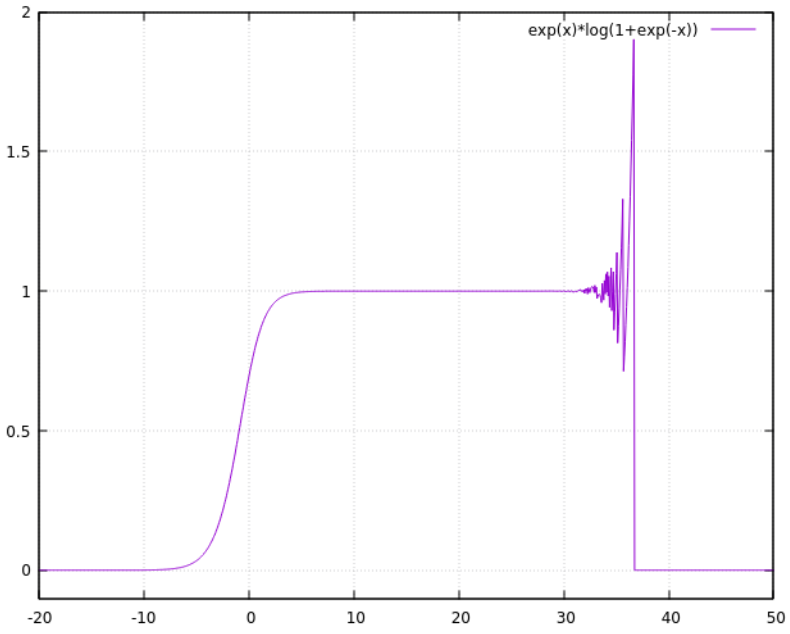
  Narysowanie wykres funkcji $f(x)=e^{x}\ln(1+e^{-x})$ w dwóch dowolnych programach do wizualizacji. Następnie policzenie granicy funkcji $\lim_{x\to\infty} f(x)$, po czym porównanie wykres funkcji z policzoną granicą.

- **Rozwiązanie**

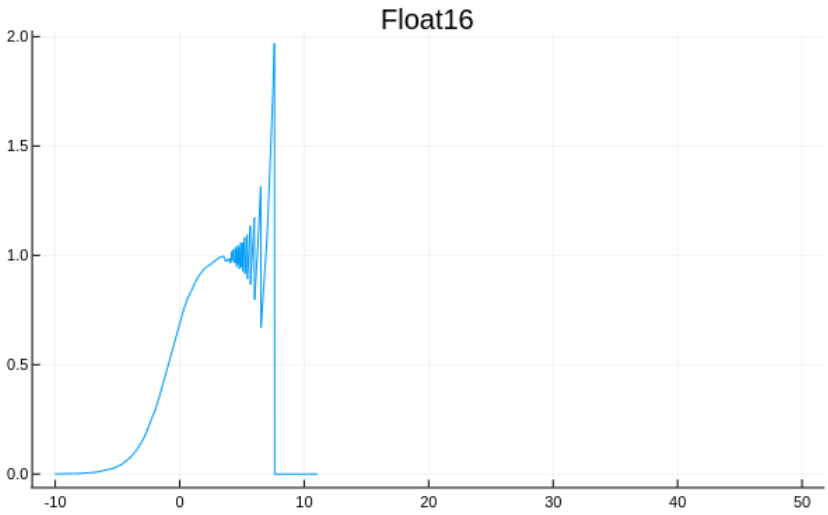
  Wykresy funkcji narywałem w programie Gnuplot oraz w języku Julia za pomocą biblioteki Plotly dla wszystkich typów zmiennopozycyjnych. Granicę funkcji obliczyłem analitycznie oraz za pomocą biblioteki SymPy w języku Julia.

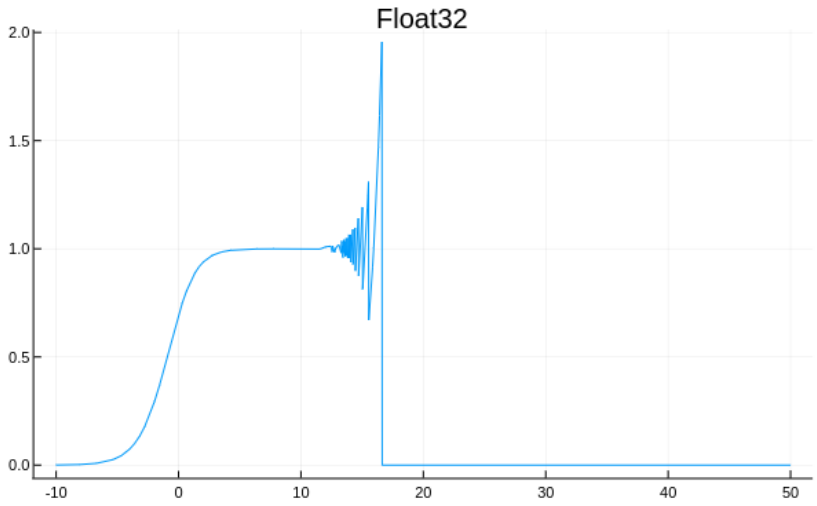
- **Wyniki**

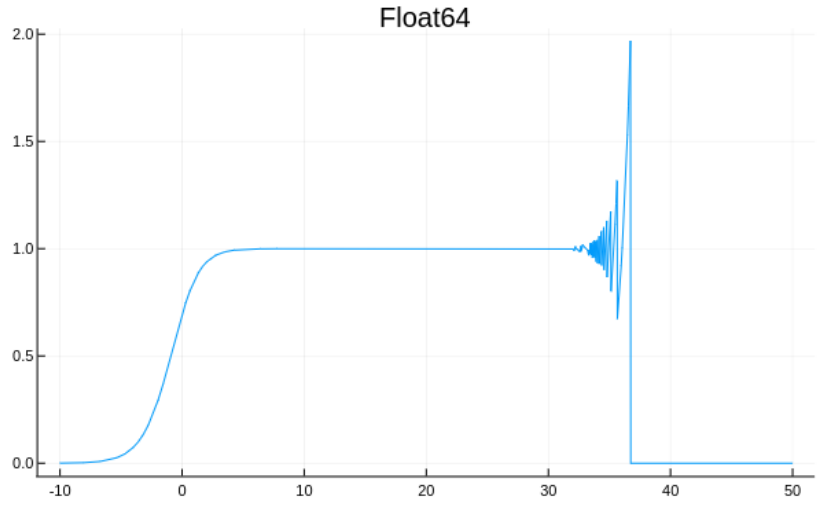
  Gnuplot:



  Plotly:







Granica obliczona za pomocą SymPy: 1.0

Granica obliczona analitycznie:

$$\lim_{x\to\infty} f(x) = \lim_{x\to\infty} e^{x}\ln\left(1+e^{-x}\right) = \lim_{x\to\infty} \frac{\ln\left(1+e^{-x}\right)}{e^{-x}} \overset{H}{=} \lim_{x\to\infty} \frac{\frac{1}{1+e^{x}}\cdot\left(-e^{-x}\right)}{-e^{-x}} = \lim_{x\to\infty} \frac{1}{1+e^{x}} = 1$$

- **Wnioski**

Granice obliczone iteracyjnie oraz za pomocą biblioteki SymPy są równe i wynoszą dokładnie jeden. Patrząc na wykresy możemy zauważyć, że do pewnego momentu funkcja faktycznie zdaje się zbiegać do jedynki, lecz po pewnym czasie pojawiają się duże niedokładności, im mniej dokładna precyzja tym dzieje się to szybciej. Błędy te wynikają z mnożenia logarytmu naturalnego którego wartość jest bardzo mała przez bardzo dużą względem niego liczbę e^{x} . Powodem zbiegania funkcji f do zera dla większych wartości x jest to, że $\ln\left(1+e^{-x}\right)\approx 0$ spowodowane jest to tym, że funkcja e^{-x} bardzo szybko zbiega do zera. W tym zadaniu małe zmiany danych spowodowały duże odchylenia od poprawnego wyniku, co oznacza, że jest ono źle uwarunkowane.

Zadanie 3

- **Opis problem**

  Rozwiązanie równania liniowego $Ax=b$ dla macierzy współczynników $A\in\mathbb{R}^{n\times n}$ i wektora prawych stron $b\in\mathbb{R}^{n}$ macierz A generowałem na 2 sposoby:

  (a) A = H_n , macierz Hilberta stopnia n,
  (b) A = R_n , macierz losowa stopnia n, z zadanym współczynnikiem uwarunkowania.

  Układ równanie należało rozwiązać za pomocą eliminacji Gaussa $x=A\backslash b$, oraz metodą inwersji $x=A^{-1}b$. Eksperyment dla macierzy Hilberta miał być przeprowadzony dla rosnącego stopnia macierzy w moim przypadku był to przedział $n\in[1,...,20]$, a dla losowej macierzy $n\in[5,10,20]$, natomiast współczynników uwarunkowania $c\in[1,10,10^3,10^7,10^{12},10^{16}]$.

- **Rozwiązanie**

  Do rozwiązania użyłem dwóch gotowych funkcji: $hilb(n)$ - generująca macierz Hilberta stopnia n, $matcond(n,c)$ - generująca losową macierz stopnia n z określonym współczynnikiem uwarunkowania c. Do tego zaimplementowałem funkcje obliczające wyniki sposobami (a) oraz (b), oraz funkcje obliczającą błąd względny dwóch wektorów.

- **Wyniki**

Macierz Hilberta				
Rozmiar	Rząd	Wskaźnik uwarunkowania	Błąd względny	
			Gauss	Inwersja
1	1	1.0	0.0	0.0
2	2	19.281470067903967	5.661048867003676e-16	1.1240151438116956e-15
3	3	524.056777586062	8.118051169482656e-15	1.7907430486334138e-14
4	4	15513.738738929262	3.349632515431573e-13	2.268815452633455e-13
5	5	476607.2502421033	2.8186181571329407e-13	3.654697861370179e-12
6	6	1.495105864172721e7	2.344229118278564e-10	2.1526924552188031e-10
7	7	4.753673559839011e8	7.410208490784287e-9	1.1004084486340423e-8
8	8	1.525757550554701e10	3.736355234341925e-7	3.044718586687223e-7
9	9	4.931538348163301e11	1.0485260733621061e-5	5.958958489267314e-6
10	10	1.6025337742793652e13	0.00019028920618170 39	0.00023929944568637062
11	10	5.219813567997335e14	0.004840612082131825	0.006720407723584139
12	11	1.6546640506383568e16	0.0515312199356675	0.06988898126789012
13	11	2.376786926717726e18	0.8773261344142039	1.5119571944035046
14	11	2.44167173619644e17	3.934776094572252	5.840670521642857
15	12	2.4022870188034854e17	35.372359774035836	31.13950131676681
16	12	7.301849062715712e17	5.345783450414308	4.99917180206163
17	12	4.7507771590947725e17	14.381303216697646	15.299960220214384
18	12	1.0621139181205069e18	5.454180864329791	6.942728611787584
19	13	4.215819618491786e18	7.007414463759739	16.272559569359128
20	13	2.365121119732083e18	11.373002753850809	16.695512931170512

Macierz Losowa				
Rozmiar	Rząd	Wskaźnik uwarunkowania	Błąd względny	
			Gauss	Inwersja
5	5	1	1.1102230246251565e-16	1.3136335981433191e-16
5	5	10	1.1102230246251565e-16	3.2177320244274193e-16
5	5	1e3	3.056934131323098e-14	1.863859131820406e-14
5	5	1e7	8.205813707143464e-11	1.1615464381382779e-10
5	5	1e12	1.5154988962728898e-5	1.6939848669805186e-5
5	4	1e16	0.6437376422550746	0.5924331280500919
10	10	1	1.0532500405730103e-16	3.1985215122904827e-16
10	10	10	1.5303368297126222e-16	3.861916815434371e-16
10	10	1e3	3.0001176165852964e-14	3.159927608870659e-14
10	10	1e7	1.4589602351368174e-10	1.145392960955392e-10
10	10	1e12	1.488824482916849e-5	1.4135325027271759e-5
10	9	1e16	0.37223274592498873	0.34120442420893987
20	20	1	4.996003610813204e-16	5.644695297351525e-16
20	20	10	3.130356458463198e-16	4.631107411915238e-16
20	20	1e3	9.049237113845489e-15	1.0539519733670682e-14
20	20	1e7	2.7793630321991436e-10	2.5191068677521605e-10
20	20	1e12	7.03479600994331e-6	7.73016210320465e-6
20	19	1e16	1.1617186238663233	1.2316767106091768

- **Wnioski**

  Patrząc na dwie powyższe tabele, można zauważyć, że niezależnie od innych wartości wzrost wskaźnika uwarunkowania macierzy jest bezpośrednio powiązany ze wzrostem błędu względnego wyniku, możemy wysnuć wnioski, że w rozwiązywaniu układów liniowych z macierzami, decydujący wpływ na poprawność wyniku ma wskaźnik uwarunkowania macierzy. Rozważając wyniki w tabeli pierwszej widzimy, że macierz Hilberta, jest bardzo niewygodna w obliczeniach, ponieważ jej wskaźnik uwarunkowania rośnie wraz z rzędem macierzy, co za tym idzie dla macierzy dużych rzędów spada precyzja obliczeń, sposobem na zwiększenie dokładności wyniku jest zastosowanie metody eliminacji Gaussa, gdyż generuje ona znacznie mniejsze błędy niż metoda inwersji.

Zadanie 4

- **Opis problemu**

  Obliczenie miejsc zerowych wielomianu Wilkinsona $P(x)$ przy użyciu funkcji roots() z pakietu Polynomials w języku Julia. Następnie, na podstawie obliczonych pierwiastków obliczenie wartości funkcji, tj. $|P(z_k)|$, $|p(z_k)|$, $|z_k - k|$, gdzie z_k to k-ty pierwiastek oraz

  $$p(x) = (x-20)(x-19)(x-18)(x-17)(x-16)(x-15)(x-14)(x-13)(x-12)(x-11)$$
  $$(x-10)(x-9)(x-8)(x-7)(x-6)(x-5)(x-4)(x-3)(x-2)(x-1)$$

$$P(x) = x^{20} - 210\,x^{19} + 20615\,x^{18} - 1256850\,x^{17} + 53327946\,x^{16} - 1672280820\,x^{15} + 40171771630\,x^{14} - 756111184500\,x^{13} + 11310276995381\,x^{12} - 135585182899530\,x^{11} + 1307535010540395\,x^{10} - 10142299865511450\,x^{9} + 63030812099294896\,x^{8} - 311333643161390640\,x^{7} + 1206647803780373360\,x^{6} - 3599979517947607200\,x^{5} + 8037811822645051776\,x^{4} - 12870931245150988800\,x^{3} + 13803759753640704000\,x^{2} - 8752948036761600000\,x + 2432902008176640000$$

Następnie należało zmienić współczynnik -210 wielomianu $P(x)$ na $-210-2^{-23}$ po czym powtórzyć eksperyment.

- **Rozwiązanie**

  Za pomocą pakietu Polynomials zaimplementowałem eksperyment.

- **Wyniki**

  (1) Tabela przed zmianą współczynnika.

k	z_k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
1	0.9999999999999771	512.0	2560.0	2.3314683517128287e-15
2	.9999999999967497	29696.0	13312.0	3.2502889268926083e-12
3	2.999999999866293	52736.0	20992.0	1.337068233908667e-10
4	4.000000011019143	681472.0	943616.0	1.1019142931445458e-8
5	4.999999748927312	5.4272e6	6.1952e6	2.510726879734193e-7
6	6.000003401569265	1.6007168e7	1.7334272e7	3.4015692653710516e-6
7	6.999966502694603	7.9131648e7	8.123904e7	3.349730539703444e-5
8	8.000248650417012	6.33514496e8	6.377088e8	0.0002486504170118309
9	8.99863071666689	1.982439424e9	1.988410368e9	0.0013692833331102605
10	10.005643046417331	7.538121216e9	7.548366848e9	0.005643046417331377
11	10.9832728819987	2.1405586432e10	2.1419172352e10	0.016727118001300667
12	12.039704742162508	5.8907896832e10	5.8929570816e10	0.039704742162507856
13	12.93541042916589	1.39891236352e11	1.3992241664e11	0.06458957083411043
14	14.086351318967639	3.61559837184e11	3.6160536064e11	0.08635131896763859
15	14.916551086146429	7.63436727296e11	7.63491395072e11	0.08344891385357123
16	16.056101927823747	1.876228873728e12	1.87629699072e12	0.05610192782374668
17	16.970222485080672	3.779840352256e12	3.779930002432e12	0.029777514919327785
18	18.009738396580197	7.442458112e12	7.442565720576e12	0.009738396580196707
19	18.99796473418517	1.4829895143936e13	1.4830035581952e13	0.002035265814829046
20	20.000189920314238	2.1896970131456e13	2.1897133972992e13	0.0001899203142379235 5

(2) Tabela po zmianie współczynnika.

k	z_k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
1	1.000000000000076 + 0.0im	8704.0	6656.0	7.593925488436071e-14
2	1.9999999999888531 + 0.0im	71680.0	55296.0	1.1146861211841497e-11
3	3.0000000005321574 + 0.0im	437760.0	345600.0	5.321574292338482e-10
4	3.9999999854321384 + 0.0im	2.80576e6	2.543616e6	1.4567861583714148e-8
5	5.000000238159716 + 0.0im	1.2812288e7	1.0252288e7	2.3815971594842722e-7
6	6.000002465516928 + 0.0im	5.4333952e7	6.712832e6	2.4655169283960277e-6
7	6.999740236517096 + 0.0im	2.20200448e8	1.124918272e9	0.00025976348290424056
8	8.006972846450305 + 0.0im	6.87242752e8	1.6825628672e10	0.006972846450304715
9	8.91828053511307 + 0.0im	2.498716672e9	1.32950144e11	0.08171946488693038
10	10.095261779402826 - 0.6421736837273344im	7.598338633861671e9	1.4764995155822578e12	0.6492009293638812
11	10.095261779402826 + 0.6421736837273344im	7.598338633861671e9	1.4764995155822578e12	1.1094765828449358
12	11.793382133721813 - 1.652045904899795im	3.452258377627477e10	3.291714323165595e13	1.6649163986703759
13	11.793382133721813 + 1.652045904899795im	3.452258377627477e10	3.291714323165595e13	2.045771821860372
14	13.992264932675088 - 2.5187943499538363im	2.6583092531771655e11	9.54413167881884e14	2.518806226891201
15	13.992264932675088 + 2.5187943499538363im	2.6583092531771655e11	9.54413167881884e14	2.712905258809403
16	16.730710830243574 - 2.812619231321066im	1.1578526902243865e12	2.7419531000493124e16	2.9059878282319693
17	16.730710830243574 + 2.812619231321066im	1.1578526902243865e12	2.7419531000493124e16	2.8254811267012934
18	19.502430503477196 - 1.9403277005007826im	7.179830058009144e12	4.2523425870966515e17	2.454010799305781
19	19.502430503477196 + 1.9403277005007826im	7.179830058009144e12	4.2523425870966515e17	2.0043223284080343
20	20.84690345245782 + 0.0im	1.241963999488e13	1.3743413755555942e18	0.8469034524578198

- **Wnioski**

  Wartości obydwóch wielomianów dla otrzymanych miejsc zerowych, są bardzo dalekie od poprawnego wyniku czyli zera (dla ostatniego pierwiastka różnica wynosi aż 10^{13}), mimo że miejsca zerowe obliczone za pomocą funkcji roots() niewiele różnią się od rzeczywistych pierwiastków, jest to spowodowane ograniczeniami arytmetyki. Patrząc na tabelę (2) możemy również zauważyć, że po zmianie jednego ze współczynników o zaledwie $2^{(-23)}$, co intuicyjnie nie powinno spowodować dużych zmian, funkcja roots() zwróciła pierwiastki zespolone, a wyniki jeszcze bardziej odbiegały od zera a odnosząc się już do samego wielomianu obliczanego w tym zadaniu, mimo że pierwiastki z funkcji roots() niewiele różnią się od rzeczywistych to wysokie potęgi w wielomianie powodują mnożenie tego błędu, co w rezultacie np. dla x^{20} daje kolosalną różnicę, po tym widać, że zadanie jest źle uwarunkowane.

Zadanie 5

- **Opis problemu**

  Zbadanie równania rekurencyjnego opisującego wzrost populacji w postaci:

  $$p_{n+1} = p_n + r p_n (1 - p_n), dla\, n = 0, 1, \ldots,$$

  gdzie r jest pewną stałą, poprzez przeprowadzenie dwóch eksperymentów (wspólne dane początkowe to $p_0 = 0.01$ i $r = 3$):

  (1) Wykonanie w Float32 40 iteracji algorytmu realizującego równanie i porównanie wyników z algorytmem po modyfikacji, polegającej na tym, że w 10 iteracji obcinamy liczbę do 3 pierwszych miejsc po przecinku.

  (2) Wykonanie 40 iteracji algorytmu realizującego równanie dla arytmetyki Float32 oraz Float64.

- **Rozwiązanie**

  Do rozwiązania tego równania zastosowałem algorytm iteracyjny, funkcja nadpisuje p na podstawie jego wcześniejszej wartości wg wzoru:

  $$p = p + r * p * (T(1.0) - p)$$

  gdzie T-precyzja arytmetyki, po czym zapisuje kolejne wartości p do tablicy. Po wykonaniu czterdziestu przejść zwracana jest tablica z rezultatami. Funkcja na wejściu przyjmuje dwie wartości:

  T – jako precyzja arytmetyki,

  cut – jako numer iteracji w której mamy uciąć liczbę, w praktyce algorytm z modyfikacja ma tą wartość ustawioną na 10, a bez mofyfikacji powinna być ona wyższa niż liczba iteracji

- **Wyniki**

Eksperyment 1		
Iteracja	Bez obcięcia	Z obcięciem
1	0.0397	0.0397
2	0.154072	0.154072
3	0.545073	0.545073
4	1.28898	1.28898
5	0.171519	0.171519
6	0.597819	0.597819
7	1.31911	1.31911
8	0.0562732	0.0562732
9	0.215593	0.215593
10	**0.722931**	**0.722**
11	1.32384	1.32415
12	0.037717	0.0364884
13	0.1466	0.141959
14	0.521926	0.50738
15	1.27048	1.25722
16	0.239548	0.287085
17	0.786043	0.901085
18	1.29058	1.16848
19	0.165525	0.577893
20	0.579904	1.30969
21	1.31075	0.0928922
22	0.0888042	0.345682
23	0.331558	1.02424
24	0.996441	0.949758
25	1.00708	1.09291
26	0.985689	0.788281
27	1.02801	1.28896
28	0.941629	0.171575
29	1.10652	0.597986
30	0.752921	1.31918
31	1.31101	0.0560039
32	0.0877831	0.214606
33	0.328015	0.720258
34	0.989278	1.32472
35	1.0211	0.0342414
36	0.956467	0.133448
37	1.08138	0.480368
38	0.817368	1.22921
39	1.2652	0.383962
40	0.258605	1.09357

Eksperyment 2		
Iteracja	Float32	Float64
1	0.0397	0.0397
2	0.154072	0.154072
3	0.545073	0.545073
4	1.28898	1.28898
5	0.171519	0.171519
6	0.597819	0.597819
7	1.31911	1.31911
8	0.0562732	0.0562716
9	0.215593	0.215587
10	0.722931	0.722914
11	1.32384	1.32384
12	0.037717	0.0376953
13	0.1466	0.146518
14	0.521926	0.521671
15	1.27048	1.27026
16	0.239548	0.240352
17	0.786043	0.788101
18	1.29058	1.28909
19	0.165525	0.171085
20	0.579904	0.596529
21	1.31075	1.31858
22	0.0888042	0.0583776
23	0.331558	0.223287
24	0.996441	0.743576
25	1.00708	1.31559
26	0.985689	0.0700353
27	1.02801	0.265426
28	0.941629	0.850352
29	1.10652	1.23211
30	0.752921	0.374146
31	1.31101	1.07663
32	0.0877831	0.829126
33	0.328015	1.25415
34	0.989278	0.297907
35	1.0211	0.925382
36	0.956467	1.13253
37	1.08138	0.682241
38	0.817368	1.33261
39	1.2652	0.00290916
40	0.258605	0.0116112

- **Wnioski**

  Analizując tabele możemy zauważyć, że obcięcie pewnej liczby cyfr znaczących, wyraźnie wpływa na wyniki dalszych obliczeń, w przypadku pierwszego eksperymentu obcięcie wykonujemy umyślnie i następuje w 10 iteracji dalej wyniki znacznie się różnią, w eksperymencie drugim obcięcie wynika z zastosowanych arytmetyk różniących się długością przechowywanych mantys, różnice wyników nie są na początku widoczne, lecz niedokładność arytmetyki Float32 w stosunku do Float64, powoduje nawarstwianie się błędu, ponieważ każda kolejna wartość zależy od poprzedniej, co od pewnego momentu również skutkuje rozbieżnością wyników. Widzimy tutaj zjawisko czułej zależności od warunków początkowych oraz sprzężenia zwrotnego. To zadanie pokazuje, że bardzo mały błąd, poprzez kolejne kroki obliczeń może się nawarstwiać i całkowicie popsuć końcowy wynik, co za tym idzie np. modelowanie komputerowe z racji ograniczonej precyzji przechowywania liczb w komputerze może nie do końca odzwierciedlać rzeczywistość którą chcemy zasymulować, możemy z tym walczyć dopierając dokładniejszą arytmetykę, lecz zawsze ilość cyfr przechowywanych w komputerze będzie skończona.

Zadanie 6

- **Opis problemu**

  Zbadanie zachowania równania rekurencyjnego:

  $$x_{n+1}=x_n^2+c \text{ dla } n=0,1,\ldots$$

  gdzie c jest pewną stałą, dla danych:

c	-2	-2	-2	-1	-1	-1	-1
x_0	1	2	1.99999999999999	1	-2	0.75	0.25

  Należało wykonać 40 iteracji ciągu, dla każdego pakietu danych i narysować wykresy.

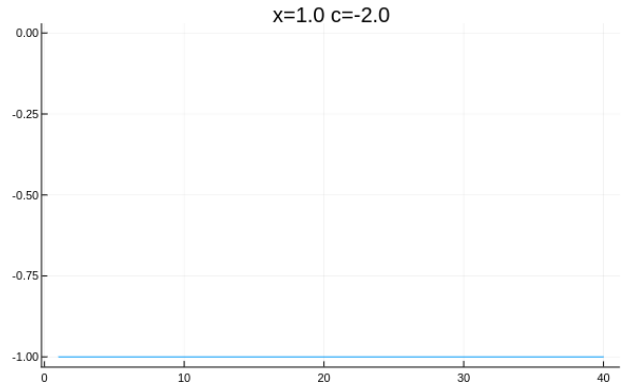
- **Rozwiązanie**

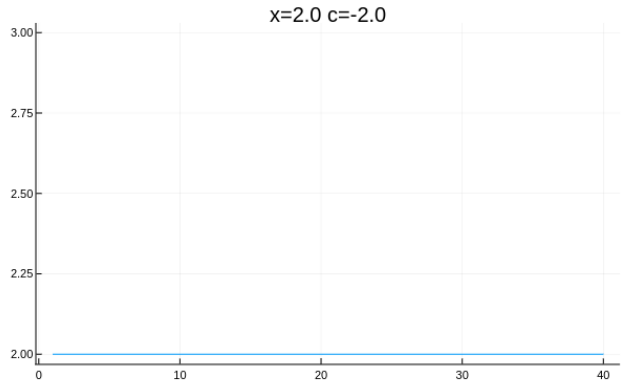
  Zaimplementowałem równanie rekurencyjne metodą iteracyjną, wpisując do tablicy kolejne wygenerowanego ciągu, na ich podstawie narysowałem wykresy, wszystko zostało policzone w arytmetyce Float64.

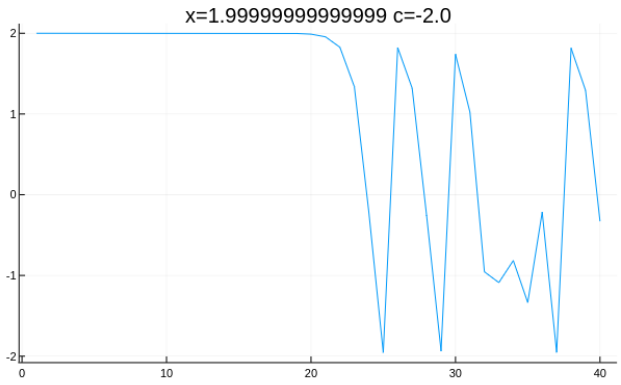
- **Wyniki**

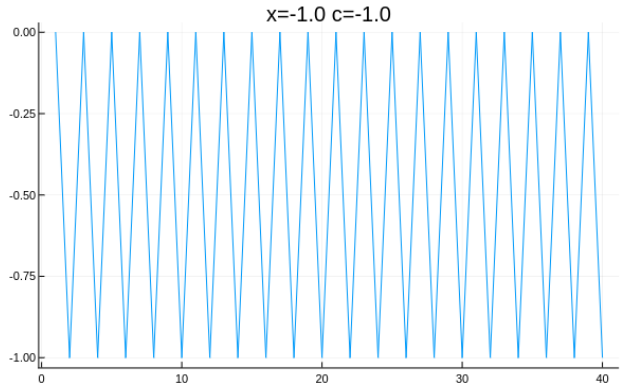
Kolejne iteracje funkcji $x_{n+1}=x_n^2+c$ dla danych $x_0,\ c$

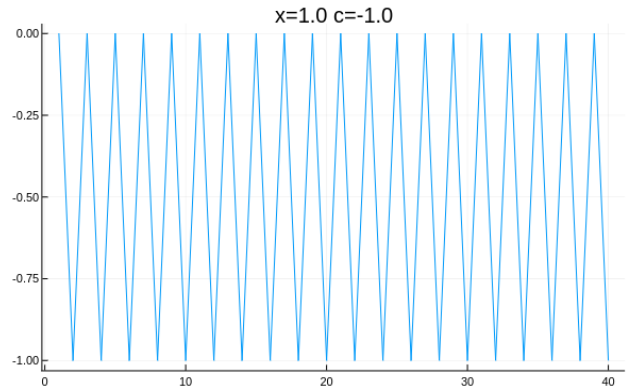
n	Dane początkowe $[x_0,\ c]$						
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
	[-2, 1]	[-2, 2]	[-2, 1.99999999999999]	[-1, 1]	[-1, -1]	[-1, 0.75]	[-1, .25]
1	-1.0	2.0	1.99999999999996	0.0	0.0	-0.4375	-0.9375
2	-1.0	2.0	1.9999999999998401	-1.0	-1.0	-0.80859375	-0.12109375
3	-1.0	2.0	1.9999999999993605	0.0	0.0	-0.3461761474609375	-0.9853363037109375
4	-1.0	2.0	1.999999999997442	-1.0	-1.0	-0.8801620749291033	-0.029112368589267135
5	-1.0	2.0	1.9999999999897682	0.0	0.0	-0.2253147218564956	-0.9991524699951226
6	-1.0	2.0	1.9999999999590727	-1.0	-1.0	-0.9492332761147301	-0.0016943417026455965
7	-1.0	2.0	1.999999999836291	0.0	0.0	-0.0989561875164966	-0.9999971292061947
8	-1.0	2.0	1.9999999993451638	-1.0	-1.0	-0.9902076729521999	-5.741579369278327e-6
9	-1.0	2.0	1.9999999973806553	0.0	0.0	-0.01948876442658909	-0.9999999999670343
10	-1.0	2.0	1.999999989522621	-1.0	-1.0	-0.999620188061125	-6.593148249578462e-11
11	-1.0	2.0	1.9999999580904841	0.0	0.0	-0.0007594796206411569	-1.0
12	-1.0	2.0	1.9999998323619383	-1.0	-1.0	-0.9999994231907058	0.0
13	-1.0	2.0	1.9999993294477814	0.0	0.0	-1.1536182557003727e-6	-1.0
14	-1.0	2.0	1.9999973177915749	-1.0	-1.0	-0.9999999999986692	0.0
15	-1.0	2.0	1.9999892711734937	0.0	0.0	-2.6616486792363503e-12	-1.0
16	-1.0	2.0	1.9999570848090826	-1.0	-1.0	-1.0	0.0
17	-1.0	2.0	1.999828341078044	0.0	0.0	0.0	-1.0
18	-1.0	2.0	1.9993133937789613	-1.0	-1.0	-1.0	0.0
19	-1.0	2.0	1.9972540465439481	0.0	0.0	0.0	-1.0
20	-1.0	2.0	1.9890237264361752	-1.0	-1.0	-1.0	0.0
21	-1.0	2.0	1.9562153843260486	0.0	0.0	0.0	-1.0
22	-1.0	2.0	1.82677862987391	-1.0	-1.0	-1.0	0.0
23	-1.0	2.0	1.3371201625639997	0.0	0.0	0.0	-1.0
24	-1.0	2.0	-0.21210967086482313	-1.0	-1.0	-1.0	0.0
25	-1.0	2.0	-1.9550094875256163	0.0	0.0	0.0	-1.0
26	-1.0	2.0	1.822062096315173	-1.0	-1.0	-1.0	0.0
27	-1.0	2.0	1.319910282828443	0.0	0.0	0.0	-1.0
28	-1.0	2.0	-0.2578368452837396	-1.0	-1.0	-1.0	0.0
29	-1.0	2.0	-1.9335201612141288	0.0	0.0	0.0	-1.0
30	-1.0	2.0	1.7385002138215109	-1.0	-1.0	-1.0	0.0
31	-1.0	2.0	1.0223829934574389	0.0	0.0	0.0	-1.0
32	-1.0	2.0	-0.9547330146890065	-1.0	-1.0	-1.0	0.0
33	-1.0	2.0	-1.0884848706628412	0.0	0.0	0.0	-1.0
34	-1.0	2.0	-0.8152006863380978	-1.0	-1.0	-1.0	0.0
35	-1.0	2.0	-1.3354478409938944	0.0	0.0	0.0	-1.0
36	-1.0	2.0	-0.2165790639847462 5	-1.0	-1.0	-1.0	0.0
37	-1.0	2.0	-1.953093509043491	0.0	0.0	0.0	-1.0
38	-1.0	2.0	1.8145742550678174	-1.0	-1.0	-1.0	0.0
39	-1.0	2.0	1.2926797271549244	0.0	0.0	0.0	-1.0
40	-1.0	2.0	-0.3289791230026702	-1.0	-1.0	-1.0	0.0

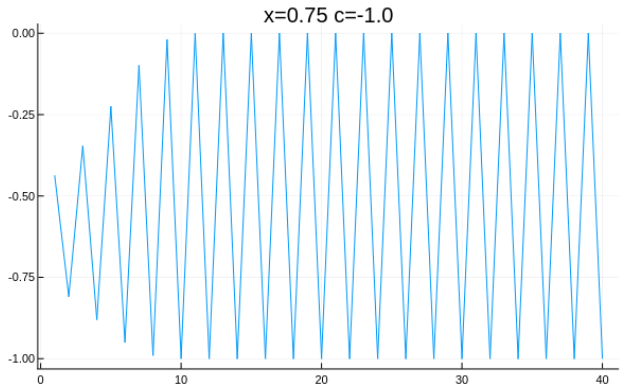

x=1.0 c=-2.0
0.00
-0.25
-0.50
-0.75
-1.00
0
10
20
30
40

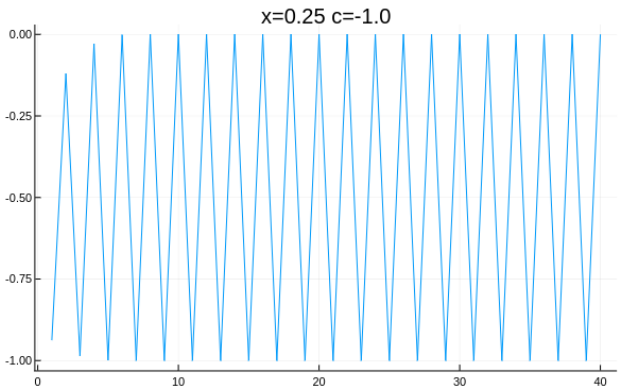

x=2.0 c=-2.0
3.00
2.75
2.50
2.25
2.00
0
10
20
30
40


x=1.99999999999999 c=-2.0
2
1
0
-1
-2
0
10
20
30
40


x=-1.0 c=-1.0
0.00
-0.25
-0.50
-0.75
-1.00
0
10
20
30
40


x=1.0 c=-1.0
0.00
-0.25
-0.50
-0.75
-1.00
0
10
20
30
40


x=0.75 c=-1.0
0.00
-0.25
-0.50
-0.75
-1.00
0
10
20
30
40


x=0.25 c=-1.0
0.00
-0.25
-0.50
-0.75
-1.00
0
10
20
30
40

- **Wnioski**

  Należy zwrócić uwagę, że to zadanie jest dosyć podobne do zadania 5, również tutaj mamy do czynienia ze zjawiskiem sprzężenia zwrotnego, choć z drugiej strony analizując kolejne tabele widzimy, że mamy tutaj do czynienia z całkiem stabilnym układem układem dla odpowiednich danych, czego nie można powiedzieć o zadaniu 5, tam bardziej pasowało by pojęcie chaosu deterministycznego. Stabilność układu określamy przez to jak jest on wrażliwy na zmianę danych początkowych. Układ wykazuje stabilność dla każdego typu danych oprócz (3), sprawdziłem również dla 100 iteracji co tylko utwierdziło mnie w przekonaniu, że dla tych danych układ nie wykazuje stabilności, chociaż różnica wartości x_0 dla zestawów (2) i (3) to tylko 0.00000000000001, widzimy dwa różne wykresy, błąd poprzez podnoszenie x_0 do kwadratu zwiększa się w kolejnych iteracjach, aż w końcu staję się na tyle duży, że wybija układ z pozycji równowagi.